Readings in VGIS, Fall 2008

Lecture 4: Hidden Markov Model

Zheng-Hua Tan

Department of Electronic Systems
Aalborg University, Denmark
zt@es.aau.dk

Part I: Markov chain

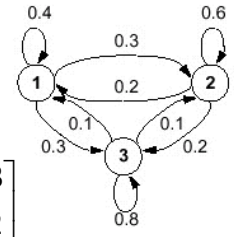
- Markov chain
- Hidden Markov model
- Basic calculations
  - How to evaluate an HMM – the forward algorithm
  - How to decode an HMM – the Viterbi algorithm
  - How to estimate HMM parameters – Baum-Welch Algorithm
- Continuous and discrete HMMs

A three-state Markov model

- The weather on day *t* is characterised by a single one of the three states

Example:A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

- The above stochastic process is considered an observable Markov model since the output process is a set of states at each instant of time, where each state corresponds to an observable event.

Basic calculations

Example: What is the probability that the weather for eight consecutive days is "sun-sun-sun-rain-rain-sun-cloudy-sun"?

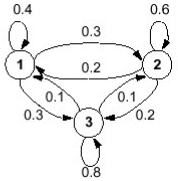
Solution:

O =	sun	sun	sun	rain	rain	sun	cloudy	sun
	3	3	3	1	1	3	2	3

$$\begin{aligned} P(\bar{O}|Model) &= P[3]P[3|3]P[3|3]P[1|3]P[1|1]P[3|1]P[2|3]P[3|2] \\ &= \pi_3 a_{33} a_{31} a_{11} a_{13} a_{32} a_{23} \\ &= 1.536 \times 10^{-4} \end{aligned}$$

Example:A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



(*After Joseph Picone*)

Log-Domain Mathematics

Multiplying many numbers together brings in the risk of underflow errors.
A solution: Transform everything into the log domain:

linear domain	*log domain*
x^y	$e^y \cdot x$
$x \cdot y$	$x+y$
$x+y$	logAdd(x,y)

logAdd(x,y) computes sum of x and y when both x and y are already in log domain.

$$\log(x+y) = \log(x+\left[\frac{y}{x}\cdot x\right])$$
$$\log(x\left[1+\frac{y}{x}\right])$$
$$\log(x)+\log(1+\frac{y}{x})$$
$$\log(x)+\log(1+e^{\log(y)-\log(x)})$$

(Hosom, 2006)

Log-Domain Mathematics

log-domain mathematics avoids underflow, allows (expensive) multiplications to be transformed to (cheap) additions.

Typically used in HMMs where there are a large number of Multiplications, ***O***(F) where F is the number of frames.

If F is moderately large (e.g. 5 seconds of speech = 500 frames), even large probabilities (e.g. 0.9) yield small results:

$0.9^{500} = 1.3\times10^{-23}$
$0.65^{500} = 2.8\times10^{-94}$

The Markov chain

- Consider a system described at any time *t* as being in one of a set of *N* states indexed by {1,2,…,*N*}
- Denote the actual state at time *t* as q_t
- So, the first-order Markov chain is

$$P[q_t = j \mid q_{t-1} = i, q_{t-2} = k, ...] = P[q_t = j \mid q_{t-1} = i]$$

- We consider those processes in which the right-side of the equation above is independent of time, leading to state-transition probabilities

$$a_{ij} = P[q_t = j \mid q_{t-1} = i], \quad 1 \le i, j \le N$$

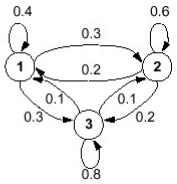
with constraints:

$$a_{ij} \ge 0 \qquad \forall j, i$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i$$

Example: A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



N=3

Part II: Hidden Markov model

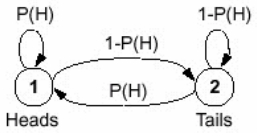
- Markov chain
- Hidden Markov model
- Basic calculations
  - How to evaluate an HMM – the forward algorithm
  - How to decode an HMM – the Viterbi algorithm
  - How to estimate HMM parameters – Baum-Welch Algorithm
- Continuous and discrete HMMs

"Hidden" Markov model

Consider the problem of predicting the outcome of a coin toss experiment. You observe the following sequence:

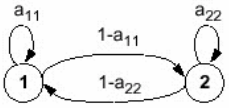
$$\bar{O} = (HHTTTHTTH\ldots H)$$

What is a reasonable model of the system?



1-Coin Model
(Observable Markov Model)

O = H H T T H T H H T T H ...
S = 1 1 2 2 1 2 1 1 2 2 1 ...



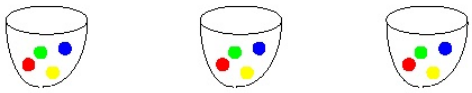
2-Coins Model
(Hidden Markov Model)

O = H H T T H T H H T T H ...
S = 2 1 1 2 2 2 1 2 2 1 2 ...

P(H) = P_1 P(H) = P_2
P(T) = 1-P_1 P(T) = 1-P_2

The Urn-and-Ball model

The Urn-and-Ball Model — doubly stochastic systems



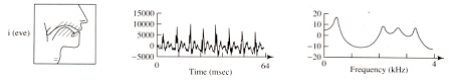
P(red) = $b_1(1)$	P(red) = $b_2(1)$	P(red) = $b_3(1)$
P(green) = $b_1(2)$	P(green) = $b_2(2)$	P(green) = $b_3(2)$
P(blue) = $b_1(3)$	P(blue) = $b_2(3)$	P(blue) = $b_3(3)$
P(yellow) = $b_1(4)$	P(yellow) = $b_2(4)$	P(yellow) = $b_3(4)$
...	...	...

Hidden State:

Suppose we can observe something that's affected by the true state.

$\bar{O}$ = {green, blue, green, yellow, red, ..., blue}

How can we determine the appropriate model for the observation sequence given the system above?



What is an HMM

- Hidden Markov Model:
  - more than 1 event associated with each state.
  - all events have some probability of emitting at each state.
  - given a sequence of outputs, we can't determine exactly the state sequence.
  - We can compute the probabilities of different state sequences given an output sequence.
- Doubly stochastic (probabilities of both emitting events and transitioning between states); exact state sequence is "hidden."

Elements of a discrete HMM

- N: the number of states
  - states, $s = \{s_1, s_2, \ldots, s_N\}$
  - state at time t, $q_t \in s$
- M: the number of observation symbols
  - observation symbols, $v = \{v_1, v_2, \ldots, v_M\}$
  - observation at time t, $o_t \in v$
- $A = \{a_{ij}\}$: state transition probability distribution
  - $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$, $1 \le i,j \le N$
- $B = \{b_j(k)\}$: observation probability distribution in state j
  - $b_j(k) = P(O_t = v_k \mid q_t = s_j)$, $1 \le j \le N$, $1 \le k \le M$
- $\pi = \{\pi_i\}$: initial state distribution
- For convenience, we use the notation: $\lambda = (A, B, \pi)$

Elements of an HMM

- **HMM is specified by:**

- states q^i

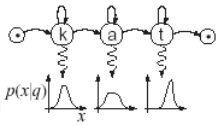
- transition probabilities a_{ij}

$$p(q_n^j | q_{n-1}^i) \equiv a_{ij}$$

	k	a	t	•
•	1.0	0.0	0.0	0.0
k	0.9	0.1	0.0	0.0
a	0.0	0.9	0.1	0.0
t	0.0	0.0	0.9	0.1

- emission distributions $b_i(x)$

$$p(x | q^i) \equiv b_i(x)$$



+ (initial state probabilities $p(q_1^i) \equiv \pi_i$)

From Dan Ellis, 2004.

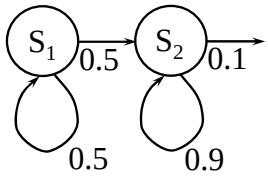
What is an HMM

an HMM still *generates* observations, (Hosom, 2006)
each state is still discrete,
observations can still come from a finite set (discrete HMMs).

- the number of items in the set of events does not have to be the same as the number of states.

- when in state *S*,
  there's $p(e_1)$ of generating event 1,
  there's $p(e_2)$ of generating event 2, etc.

p_{S1}(black) = 0.3
p_{S1}(white) = 0.7



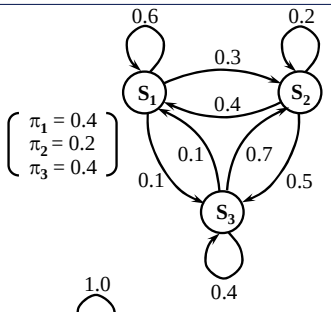
p_{S2}(black) = 0.6
p_{S2}(white) = 0.4

What is an HMM

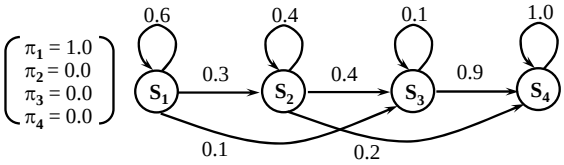
- must know all possible states in advance
- must know possible state connections in advance
- cannot recognize things outside of model
- must have some estimate of state emission probabilities and state transition probabilities
- make several assumptions (usually so math is easier)
- if we can find best state sequence through an HMM for a given observation, we can compare multiple HMMs for recognition.

HMM Topologies

Ergodic (fully-connected)



Bakis (left-to-right)



•Topology defined by the state transition matrix (If an element of this matrix is zero, there is no transition between those two states).
•The topology must be specified in advance by the system designer

Part III: Three HMM calculations

- Markov chain
- Hidden Markov model
- Basic calculations
  - How to evaluate an HMM – the forward algorithm
  - How to decode an HMM – the Viterbi algorithm
  - How to estimate HMM parameters – Baum-Welch Algorithm
- Continuous and discrete HMMs

Three basic HMM problems

1. **Scoring:** Given an observation sequence $\boldsymbol{O} = \{o_1,o_2,...,o_T\}$ and a model $\lambda = \{\boldsymbol{A}, \boldsymbol{B},\pi\}$, how to compute $P(\boldsymbol{O} \mid \lambda)$, the probability of the observation sequence? → The Forward-Backward Algorithm

2. **Matching:** Given an observation sequence $\boldsymbol{O} = \{o_1,o_2,...,o_T\}$ and the model λ,how to choose a state sequence $\boldsymbol{q} = \{q_1,q_2,...,q_T\}$ which is optimum in some sense? → The Viterbi Algorithm

3. **Training:** How to adjust the model parameters $\lambda = \{A,B,\pi\}$ to maximize $P(\boldsymbol{O}| \lambda)$? → The Baum-Welch Re-estimation Procedures

Problem 1: Scoring

- Given $\boldsymbol{O} = \{o_1, o_2, \ldots, o_T\}$ and $\lambda = \{\boldsymbol{A}, \boldsymbol{B}, \pi\}$, how to compute $P(\boldsymbol{O} \mid \lambda)$, the probability of the observation sequence? (probability evaluation)
  - Consider all possible state sequences (N^T) of length T:

$$P(O \mid \lambda) = \sum_{\text{all } \mathbf{q}} P(O \mid q, \lambda) P(q \mid \lambda)$$
$$= \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

- Calculation required $\approx N^T \cdot 2T$
  - For N =5,T = 100, $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations!

The forward algorithm

- Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i \mid \lambda)$$

i.e., the probability of the partial observation sequence until time t and state i at time t, given the model λ

- We can solve for $\alpha_t(i)$ inductively as follows:

1. Initialisation $\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$

2. Induction $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \quad 1 \le t \le T-1, \quad 1 \le j \le N$

3. Termination $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

- Calculation$\approx T \cdot N^2$. For N=5,T=100, 2500, instead of 10^{72}

Illustration of forward algorithm

From (Rabiner, 1989)

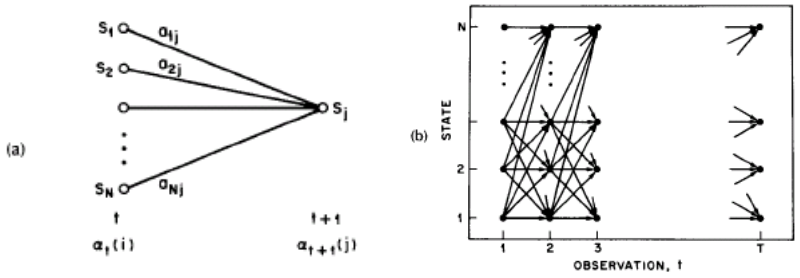


Fig. 4. (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations t, and states i.

The backward algorithm

- Similarly, consider the backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid q_t = i, \lambda)$$

i.e., the probability of the partial observation sequence from time t +1 to the end, given state i at time t and model λ

- We can solve for $\beta_t(i)$ inductively as follows:

1. Initialisation

$$\beta_T(i) = 1, \qquad 1 \le i \le N$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad t = T-1, T-2, ..., 1, \quad 1 \le i \le N$$

3. Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)$$

- Again, calculation≈$N^2 \cdot T$.

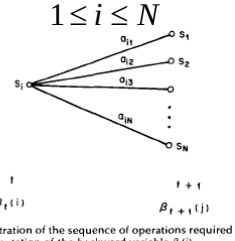
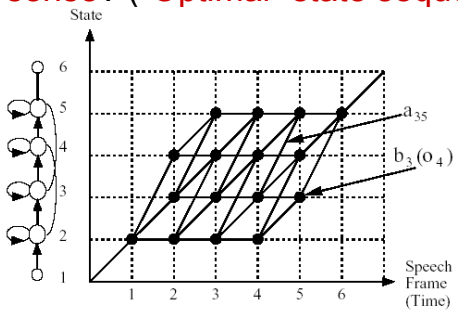


Fig. 5. Illustration of the sequence of operations required for the computation of the backward variable $\beta_t(i)$.

Problem 2: Matching

- Given $\boldsymbol{O} = \{o_1, o_2, \ldots, o_T\}$, how to choose a state sequence $\boldsymbol{q} = \{q_1, q_2, \ldots, q_T\}$ which is optimum in some sense? ("Optimal" state sequence)



Trellis diagram for an Isolated Word Recognition task. From (Young et al. 1997, p. 10)

Finding optimal state sequence

- One optimality criterion is to choose the states q_i that are individually most likely at each time t
  - Define the probability of being in state i at time t, given the observation sequence O, and the model λ

$$\gamma_t(i) = P(q_t = i \mid O, \lambda) = \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)} = \frac{P(O, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)}$$

Since $P(O, q_t = i \mid \lambda) = \alpha_t(i)\beta_t(i)$

We have $\gamma_t(i) = \dfrac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$

  - The individually most likely state q_t^* at time t is

$$q_t^* = \arg\max_{1 \le i \le N} [\gamma_t(i)]$$

Finding optimal state sequence (cont'd)

- The individual optimality criterion has the problem that the optimum state sequence may not obey state transition constraints →

  The "optimal" state sequence may not even be a valid sequence (a_{ij}=0 for some i and j)
- Another optimality criterion is is to find the single best state sequence (path), i.e., to maximize $P(\boldsymbol{q}, \boldsymbol{O}|\lambda)$ →

  The Viterbi algorithm – a method based on dynamic programming

The Viterbi algorithm

- To find the best path $\boldsymbol{q} = \{q_1, q_2, \ldots, q_T\}$, for given $\boldsymbol{O} = \{o_1, o_2, \ldots, o_T\}$, we define the best score (highest probability) along a single path, at time t,

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t \mid \lambda)$$

which accounts for the first t observations and ends in state i.

Then $$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] . b_j(o_{t+1})$$

The Viterbi algorithm (cont'd)

1. Initialisation $\delta_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$

   $\psi_1(i) = 0$

2. Recursion

   $\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] b_j(o_t), \quad 2 \le t \le T \quad 1 \le j \le N$

   $\psi_t(j) = \arg\max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}], \quad 2 \le t \le T \quad 1 \le j \le N$

3. Termination

   $P^* = \max_{1 \le i \le N}[\delta_T(i)]$

   $q_T^* = \arg\max_{1 \le i \le N}[\delta_T(i)]$

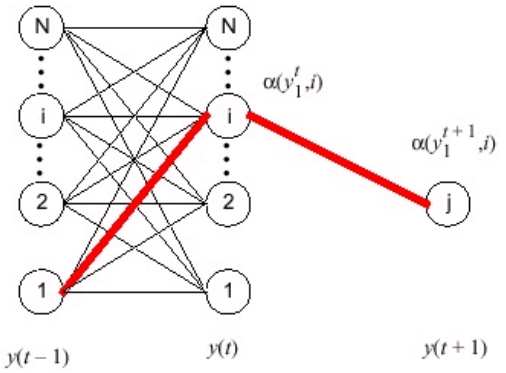
4. Path (state sequence) backtracking

   $q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1$

The Viterbi algorithm (cont'd)

From Joseph Picone

The *Viterbi algorithm* can also be used to find the best state sequence. Note that the principal difference is that we model the overall sequence probability by the probability of the single best path:



The power of recursive equation

Computing factorials *n!*

Method 1. simply caculate $n!$ for each n

Method 2. use $n! = n(n-1)!$

if $F(n) = n!$ then

$F(n) = nF(n-1)$ for $n \geq 1$

(Recursive Equation)

Problem 3: Training

- How to tune the model parameters $\lambda = \{A, B, \pi\}$ to maximize $P(\boldsymbol{O}|\lambda)$? - a learning problem
  - No efficient algorithm for global optimisation
  - Effective iterative algorithm for local optimisation: the Baum-Welch re-estimation
- Baum-Welch
  - = forward-backward algorithm (Baum, 1972)
  - is a special case of EM (expectation-maximization) algorithm
  - computes probabilities using current model λ;
  - refines λ to $\bar{\lambda}$ such that $P(\boldsymbol{O}|\lambda)$ is locally maximised
  - uses α and β from forward-backward algorithm

Baum-Welch re-estimation

Define $\xi_t(i,j)$, the probability of being in state *i* at time *t*, and state *j* at time *t*+1, given λ and ***O***, i.e.

$$\begin{aligned}\xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid \mathbf{O}, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, \mathbf{O} \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}\end{aligned}$$

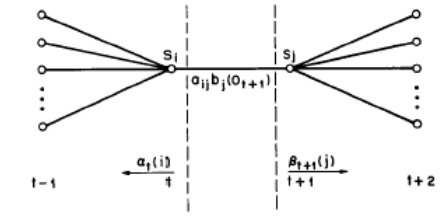


Fig. 6. Illustration of the sequence of operations required for the computation of the joint event that the system is in state S_i at time t and state S_j at time $t + 1$.

Baum-Welch Re-estimation (cont'd)

- Recall that $\gamma_t(i)$ is defined as the probability of being in state i at time t, given the entire observation sequence and the model, so

$$\gamma_t(i) = P(q_t = i \mid O, \lambda) = \sum_{j=1}^{N} P(q_t = i, q_{t+1} = j \mid O, \lambda) = \sum_{j=1}^{N} \xi_t(i,j)$$

- Sum $\gamma_t(i)$ and $\xi_t(i,j)$ over *t*, we have

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i \text{ in } \mathbf{O}$$

$$(\sum_{t=1}^{T} \gamma_t(i) = \text{the expected number of times that state } i \text{ is visitied.})$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from state } i \text{ to state } j \text{ in } \mathbf{O}$$

Baum-Welch re-estimation formulas

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } i \text{ at time } (t=1) = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \overset{s.t. o_t = v_k}{=} \frac{\sum_{t=1}^{T} P(O, q_t = i \mid \lambda) \cdot \delta(o_t, v_k)}{\sum_{t=1}^{T} P(O, q_t = i \mid \lambda)}$$

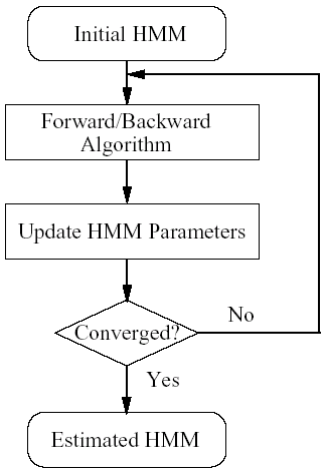
$$\delta(o_t, v_k) = \begin{cases} 1 & o_t = v_k \\ 0 & \text{otherwise} \end{cases}$$

Parameter re-estimation process

1. Initialize $\lambda = \{A, B\}$
2. Compute $\alpha, \beta,$ and ξ
3. Estimate $\bar{\lambda} = \{\bar{A}, \bar{B}\}$ from ξ
4. Replace λ with $\bar{\lambda}$
5. If not converged go to 2

It can be shown that

$$P(O \mid \bar{\lambda}) > P(O \mid \lambda) \text{ unless } \bar{\lambda} = \lambda$$



Basic operations in HMMs

For an observation sequence $O = O_1 \ldots O_T$, the three basic HMM operations are:

Problem	Algorithm	Complexity
Evaluation: Calculating $P(q_t=S_i \mid O_1 O_2 \ldots O_t)$	Forward-Backward	$O(TN^2)$
Inference: Computing $Q^* = \text{argmax}_Q P(Q\mid O)$	Viterbi Decoding	$O(TN^2)$
Learning: Computing $\lambda^* = \text{argmax}_\lambda P(O\mid\lambda)$	Baum-Welch (EM)	$O(TN^2)$

T = # timesteps, N = # states

(Andrew Moore)

Part VI: Continuous and discrete HMMS

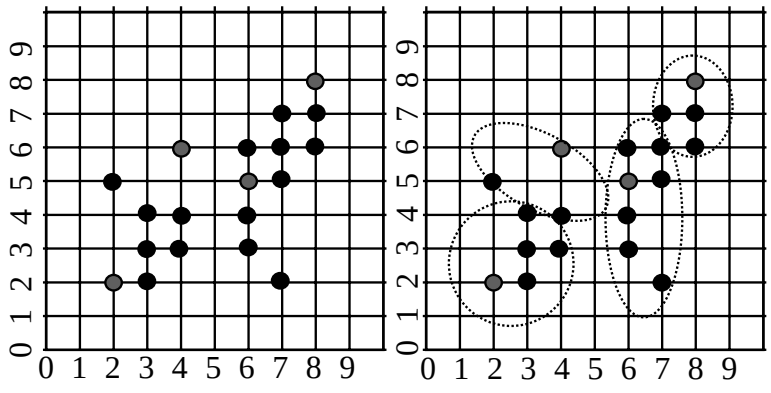
- Markov chain
- Hidden Markov model
- Basic calculations
  - How to evaluate an HMM – the forward algorithm
  - How to decode an HMM – the Viterbi algorithm
  - How to estimate HMM parameters – Baum-Welch Algorithm
- Continuous and discrete HMMs

Vector quantization

- Vector Quantization (VQ) is a method of automatically partitioning a feature space into different clusters based on training data.
- Given a test point (vector) from the feature space, we can determine the cluster that this point should be associated with.
- A "codebook" lists central locations of each cluster, and gives each cluster a name (usually a numerical index).
- This can be used for data reduction (mapping a large number of feature points to a much smaller number of clusters), or for probability estimation.
- Requires data to train on, a distance measure, and test data.

Vector quantization – an example

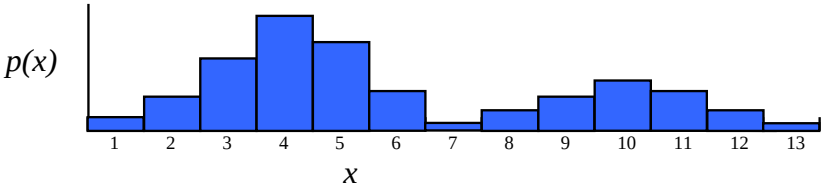
Given the following data points, create codebook of 4 clusters, with initial code word values at (2,2), (4,6), (6,5), and (8,8)



(Hosom, 2006)

Vector quantization

- Features ⇔ observations, probability of feature = $b_j(\mathbf{o}_t)$
- However, quantization error can arise when modeling a continuous signal (feature space) with discrete units (clusters)

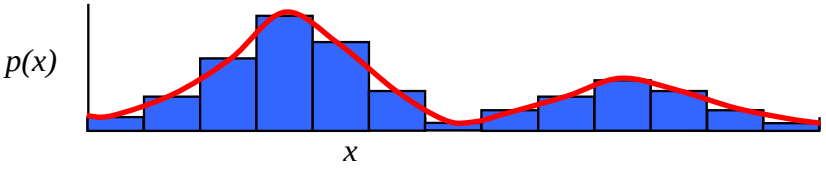


- What happens to $p(x)$ if feature space moves back and forth between bins 3 and 4? What about between bins 5 and 6?
- In addition, initialization can influence the location and histogram counts of the final clusters… want more robustness

(Hosom, 2006)

Continuous probability distribution

- What we want is a smooth, robust estimate of $p(x)$ (and $b_j(\mathbf{o}_t)$)!!
- How about this:

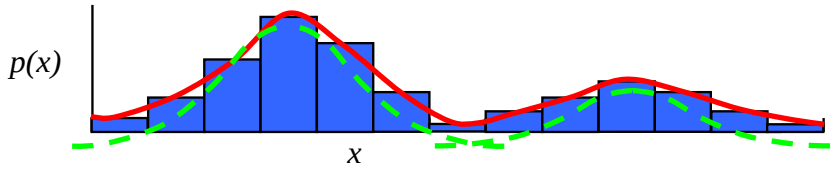


- Now, small movement along x axis has smooth, gradual effect on $p(x)$.
- Still a question about initialization…

(Hosom, 2006)

Continuous probability distribution

- One way of creating such a smooth model is to use a mixture of Gaussian probability density functions (p.d.f.s).
- The detail of the model is related to the number of Gaussian components
- This Gaussian Mixture Model (GMM) is characterized by
  (a) the number of components,
  (b) the mean and standard deviation of each component,
  (c) the weight (height) of each component

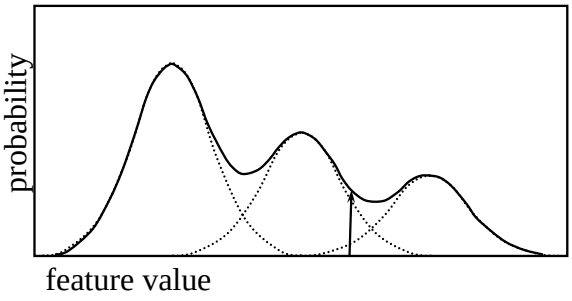


- One remaining question: how to compute probabilities from p.d.f. at one point (a single *x* value)

(Hosom, 2006)

Gaussian mixture models

- Typical HMMs for speech are continuous-density HMMs
- Use Gaussian Mixture Models (GMMs) to estimate probability of "emitting" each observation given the speech category (state).



- Features ⇔ observations, probability of feature = $b_j(\mathbf{o}_t)$

(Hosom, 2006)

Continuous density HMMs

- Replaces the discrete observation probabilities, $b_j(k)$, by a continuous PDF (probability density function) $b_j(x)$
- The PDF $b_j(x)$ is often represented as a mixture of Gaussians:

c_{jk} is the mixture weight, $c_{jk} \geq 0$, and $\sum_{k=1}^{M} c_{jk} = 1$

$$b_j(\boldsymbol{x}) = \sum_{k=1}^{M} c_{jk} N[\boldsymbol{x}, \mu_{jk}, \Sigma_{jk}] \qquad 1 \leq j \leq N$$

N is the normal density

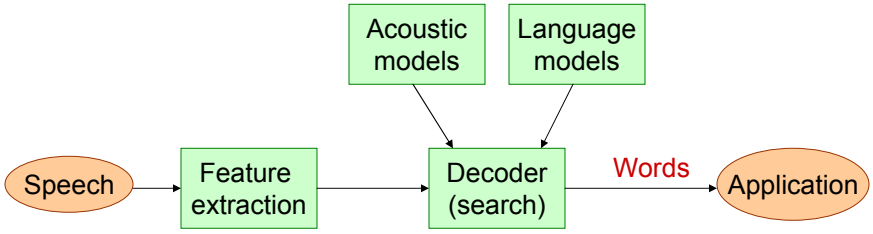
The mean and covariance matrix associated with state j and mixture k

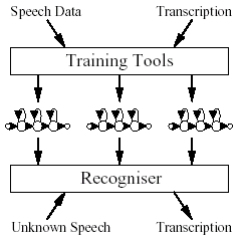
Continuous vs discrete HMMs

Comparing continuous (GMM) and discrete (VQ) HMMs:

- Continuous HMMs:
  - assume independence of features for diagonal matrix
  - require large number of components to represent arbitrary function
  - large number of parameters = slow, can't always train well
  - small number of components may not represent speech well
- Discrete HMMs:
  - quantization errors at boundaries
  - relies on how well VQ partitions the space
  - sometimes problems estimating probabilities when unusual input vector not seen in training

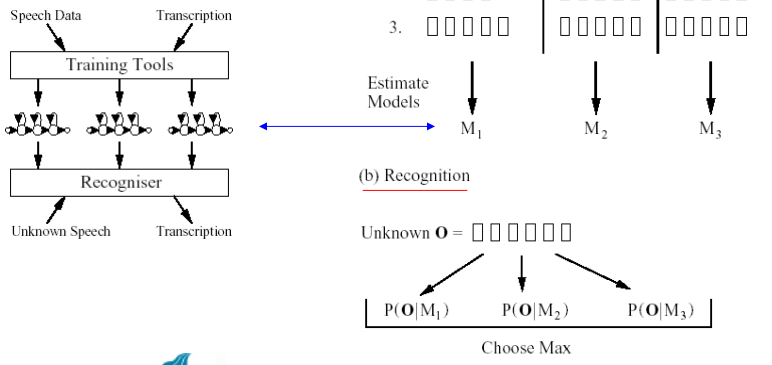
Speech recognition system





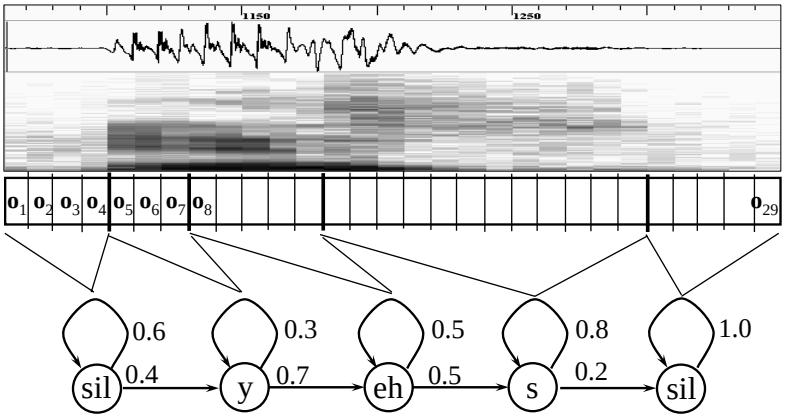
Training and test procedures for IWR

From (Young et al. 1996)



HMMs for speech

• Example of using HMM for word “yes” on an utterance:



$b_{sil}(\mathbf{o}_1)\cdot 0.6\cdot b_{sil}(\mathbf{o}_2)\cdot 0.6\cdot b_{sil}(\mathbf{o}_3)\cdot 0.6\cdot b_{sil}(\mathbf{o}_4)\cdot 0.4\cdot b_{y}(\mathbf{o}_5)\cdot 0.3\cdot b_{y}(\mathbf{o}_6)\cdot 0.3\cdot b_{y}(\mathbf{o}_7)\cdot 0.7$ ...

observation — state

(Hosom, 2006)

Summary

- Markov chain
- Hidden Markov model
- Basic calculations
  - How to evaluate an HMM – the forward algorithm
  - How to decode an HMM – the Viterbi algorithm
  - How to estimate HMM parameters – Baum-Welch Algorithm
- Continuous and discrete HMMs